Interpreting Natural Language Queries using the UMLS

Stephen B. Johnson, Ph.D., Anthony Aguirre, M.S., M.L.S., Ping Peng, Ph.D., James Cimino, M.D.

Center for Medical Informatics
Columbia Presbyterian Medical Center
New York, NY 10032

This paper describes AQUA (A QUery Analyzer), the natural language front end of a prototype information retrieval system. AQUA translates a user's natural language query into a representation in the Conceptual Graph formalism. The graph is then used by subsequent components to search various resources such as databases of the medical literature. The focus of the parsing method is on semantics rather than syntax, with semantic restrictions being provided by the UMLS Semantic Net. The intent of the approach is to provide a method that can be emulated easily in applications that require simple natural language interfaces.

INTRODUCTION

There is a rapidly growing number of on-line resources with information vital to medical practice, research, and education. A problem of paramount importance to the Medical Informatics community is how to provide access to this wealth of information and knowledge in a manner that has minimal impact on time spent by users. Most biomedical professionals have little trouble learning to query a single information resource if the information they seek is sufficiently important. However, when many resources are available, the prospect of learning to use several different query interfaces is not practical. An alternative approach is to provide an interface that accepts natural language input, freeing the user from the constraints of a formal query language. The natural language interface serves as a uniform front-end in which users can state their information needs, which can then be translated into the protocols required by various databases and knowledge bases.

Improvements in software development environments, increases in efficiency of high-level languages, and advances in the science of natural language processing have made such interfaces a practical reality [1, 2]. There are several commercial products that provide natural language interfaces to relational database systems such as Natural Language Inc., and IBM's Language Access. The feasibility of this technology is due to the fact that it is only necessary to obtain a representation of the concepts in which the user is interested and the ways in which these concepts may be semantically related; deep understanding of the query through knowledge-based techniques is not required.

The AQUA parser is built from several components that are either emerging standards or very widely used: Definite Clause Grammar (DCG) [3], Conceptual Graphs (CG) [4], and the Unified Medical Language System (UMLS) Semantic Net [5]. These three formalisms can be combined in a natural manner: the grammar specifies how phrases representing medical concepts and relationships are ordered in sentences; the Semantic Net prescribes which concepts and relations can sensibly combine; and CG provides a convenient graphical representation of concepts and relations, and of the resulting interpretation of the user's query. The intent of the approach is to provide a method that can be emulated easily in applications that require simple natural language interfaces.

METHODS

Lexicon

A Lexicon was built with the aid of the UMLS Metathesaurus. Words were assigned semantic types automatically using techniques that infer semantic types of words based on the semantic types of Metathesaurus terms. These assignments were subjected to manual review, and words lacking any assignment were classified by hand.

In addition, the Lexicon identifies possible semantic roles a word may play in a query: argument, operator, or English (a word may play more than one role). The semantic role identifies whether a word is general English, or medical. If medical, the role field identifies whether a word can function as an argument, or an operator. If the word is an argument, the last field gives its UMLS semantic type; if an operator, this field gives its semantic relation.

The Lexicon also supplies information about the syntactic class of a word (noun, verb, adjective, preposition, etc.), although this information is seldom needed by the parser. When there are multiple morphologic variants of a word which are formed regularly (e.g., *treat, treats, treated, treating*), only the root form is included in the lexicon.

```
query(G) --> query_sentence(G1),
  conjoined_query(G1,G).
conjoined_query(G1,G) --> query_sentence(G2),
  {combine_graphs(G1,G2,G3)},
  conjoined_query(G3,G).
conjoined_query(G,G) --> [].
query_sentence(G) --> pre_sentence,
  compound_sentence(G), post_sentence.
compound_sentence(G) --> sentence(G1),
  conjoined_sentence(G1,G).
conjoined_sentence(G1,G) --> conjunction,
  sentence(G2), {combine_graphs(G1,G2,G3)},
  conjoined_sentence(G3,G).
conjoined_sentence(G,G) --> [].
sentence(G) --> arg_phrase(G).
sentence(G) --> operator(R), arg_phrase(G1),
  arg_phrase(G2), {relate_graphs(G1,R,G2,G)}.
arg_phrase(G) --> argument(G1),
  predicate(G1,G).
predicate(G1,G) --> optional_conj, relative,
  operator(R), optional_prep, arg_phrase(G2),
  {relate_graphs(G1,R,G2,G3)}, predicate(G3,G).
predicate(G,G) --> [].
argument(G) --> term(G1), conjoined_arg(G1,G).
conjoined_arg(G1,G) --> conjunction, term(G2),
  {join_graphs(G1,G2,G)}, conjoined_arg.
conjoined_arg(G,G) --> [].
```

Figure 1. DCG for User Queries (Simplified)

Definite Clause Grammar

DCG is a widely used notation for grammar development that comes packaged with most implementations of the Prolog language. A grammar is a collection of rules that defines a set of sequences of symbols (in this case, the words employed in user queries). Consider the first rule of the small DCG shown in Figure 1. This rule states that a sequence of words entered by the user can be parsed as a **query** if an initial portion of the word sequence can be parsed as a **query_sentence**, and the remaining portion as a **conjoined_query** (subsequent sentences of the query). The third rule states that **conjoined_query** may be null (consumes no words from the input).

Attributes may be passed from one rule to another in order to test constraints and build an interpretation of the input. In AQUA, the only attributes passed are conceptual graphs, which are interpretations of various portions of the user's query. In the first rule of Figure 1, the word sequence parsed by **query_sentence** has an interpretation represented by the attribute G1. This attribute is passed to **conjoined_query**, which parses the remainder of the user's query and returns the resulting interpretation in attribute G (which incorporates G1). The attribute G is passed back to **query** as an interpretation of the whole user query.

A DCG may be augmented with restrictions that appear in curly brackets anywhere on the right side of a rule. For example, the second rule in Figure 1 has a restriction **combine_graphs**, which requires that the attributes G1 and G2 (conceptual graphs representing the two preceding sentences) can be combined to form a composite graph G3. Restrictions have all the power of a Prolog program, but in AQUA they are limited to a small number of operations on conceptual graphs. (These are described in greater detail below).

DCG is executed as a top-down, backtracking parser, using Prolog's native control strategy. AQUA's grammar is designed to be as deterministic as possible, i.e., it avoids the need to backup when what follows in a query is not what is expected by a grammar rule. Rules are designed so that, in general, it is possible to determine quickly whether the first item on the right hand side of the rule is present in the input. In this way, a given rule can be rejected quickly if it does not apply, or committed to if it does, eliminating other possible choices.

Semantic restrictions are applied as soon as possible: non-semantic words (e.g., *the*) are immediately distinguished from words having medical meaning; terms connected by a conjunction word (e.g., *and*) are immediately tested to see if they are of compatible semantic types; terms linked by a relationship word (e.g., a verb) are immediately tested to determine whether a semantic relation can link the two terms. In this manner, incorrect paths through the rules of the grammar are rejected as early as possible.

Operator Grammar

Interpretation of user queries is more difficult in some respects than analysis of sublanguage texts (e.g., chest x-ray reports), because the amount of variation in syntactic structures is very high (many ways to say the same thing), and because the number of medical terms that may occur is much larger (essentially anything in medicine). User queries are simpler than completely unconstrained English because the user's task is highly focused, namely to get information out of some resource, such as a database of literature. Texts in very constrained domains can be interpreted successfully using

Pattern	Examples
Arg OP Arg	Patient WITH Liver Abscess.
	Multiple Sclerosis DUE to Mycoplasmas.
	Patient who HAS Hodgkin's Disease.
	Sarcoma ARISING in Teratoma.
	Traumatic Pressure INJURIES to Sciatic Nerve.
OP Arg Arg	USE of Chloral Hydrate in Pediatric Patients.
	APPLICATION of Computers in the Intensive Care Unit.
Arg Arg	Cancer Patient.
	Head Injury.
Arg and Arg	Sedatives and Memory.
	Vitamin C and Immunity.

Figure 2. Operator-Argument Patterns

semantic grammars [6], which use almost no syntactic information, except for the ordering and nesting of semantic units. In texts with broader coverage, e.g. patient histories, significantly more syntactic information is necessary to avoid misinterpreting or ignoring important medical information [7].

The grammar rules of AQUA are based on the linguistic theory of Operator Grammar [8-9], which falls somewhere in between the syntactic and semantic extremes. Operator Grammar divides words into operators and arguments, generalizing over conventional syntactic distinctions (e.g., the difference between verbs, prepositions, adjectives, and nominalizations). In Operator Grammar, syntax is viewed as different ways of ordering operators and arguments, just as arithmetic expressions may be written in prefix, postfix, or infix notation. Unlike arithmetic, natural language is further complicated by modifiers (e.g., adverbs), additional 'marker' words (e.g., certain prepositions preceding arguments), endings on words, and the ability to omit words in contexts in which they can be reconstructed.

Consider the patterns of operators and arguments in Figure 2. In the first sentence, the preposition *with* links *patient* to *liver Abscess*. The other sentences in this group show that the same pattern applies whether the operator is an adjective, verb, or noun. In the second group of sentences, the operator is followed by its arguments. In the first sentence of this group, the operator *use* comes first, and is followed by its two arguments which are preceded by 'marker' words *of* and *in*, respectively. In the third and fourth groups, there is an implicit relationship between the two arguments. For example, *cancer patient* could be paraphrased as *cancer occurs in patient.*

Operator Grammar relates the surface appearance of sentences more directly to their meaning, avoiding the complex mapping from syntactic representations to semantic representations required by conventional syntactic approaches. The DCG in Figure 1 demonstrates some of the concepts of Operator Grammar. The structure **sentence** is the principal unit of information. The actual grammar contains several rules for **sentence** to reflect the different orderings of operators and arguments, but only the two most common are shown here. **Arg_phrase** consists of a medical concept (**argument**), optionally followed by one or more **predicates** that semantically relate the concept to other concepts.

There are several kinds of **predicate**, but the most common consists of an operator followed by an argument. The operator may be preceded by an conjunction (e.g., *and*) or relative pronoun (e.g., *which*). This single rule will match a wide variety of syntactic structures: the verb and object of a sentence, a prepositional phrase, a relative clause, or a nominalization (see Figure 2 for examples). **Argument** will parse a sequence of medical terms linked by conjunctions (e.g., comma). The details of **term** are not shown, but its function is to recognize multi-word medical terms, e.g. *intensive care unit, non-barbituate sedative hypnotic*.

Skipping

One of the ways in which AQUA differs from a syntactic parser is its ability to skip over non-essential portions of a user's query. For example, the rule for **pre_sentence** describes how to skip over initial phrases that users employ to frame their requests, e.g. "*I am interested in articles about ....*" **Post_sentence** performs a similar function at the ends of queries, e.g. "*... is topic of interest*".

When all options of a rule fail, a different method of skipping may be used to move ahead in the sentence to a point where parsing can be resumed. This technique overcomes some of the limitations of the top-down control strategy. A drastic example is skipping a sentence in a multi-sentence query. The rules in Figure 3 can be added after the **query_sentence** rule. If all other choices for this rule fail, the **skip_sentence** rule will skip words until a

```
query_sentence([]) --> skip_sentence.
skip_sentence --> terminator.
skip_sentence --> word, skip_sentence.
```

Figure 3. Skipping Rules

terminator (e.g., period) is encountered in the input. Parsing can then continue with the next sentence.

Conceptual Graphs

Conceptual Graphs (CG) are an emerging standard for knowledge representation (ANSI X3H4). CG is intended to be an easily readable form of predicate calculus, and is especially suited to the representation of natural language semantics. The parser described here uses only a small subset of the CG formalism, essentially that required to represent a semantic net with inheritance.

In the linear notation for CG, concepts are placed in square brackets. A concept may be followed by a dash and then a sequence of one or more conceptual relations. Each relation appears in parentheses and is followed by an arrow, which is followed by the related concept. The direction of the arrow indicates how to read the relation from the graph, i.e., from left to right or right to left. Consider the following user query taken from the NLM test collection: "Request search for papers detailing infections, specifically liver abscesses, in patients with Hodgkin's disease." The graph generated by AQUA is shown in Figure 4.

```
[Pathologic Function: {infections, liver
    abscesses} ] -
    (occurs_in)->[Patient or Disabled Group :
    patients ] -
        (occurs_in)<-[Disease or Syndrome :
        Hodgkin's disease].
```

Figure 4. Query Graph

The semantic relations of this graph may be read as: "pathologic function occurs in patient or disabled group which has occurrence disease or syndrome." A conceptual graph can be "oriented" in different ways by choosing any of its concepts as the main concept. Note that the parser places the specific words employed by the user after the semantic type, separated by a colon. This list of phrases is called the "referent" of the concept. The referents of concepts in the graph will be used in subsequent phases of processing to create a search against an information resource. For example, the phrases could be mapped to the Medical Subject Headings (MeSH) used to index MEDLINE.

With only a slight variation in notation, CG can be entered, processed, and displayed using the Prolog language. This feature made integration of CG with DCG quite natural. Constraints in the grammar are implemented by attempting to construct a composite graph from the graphs of two related structures. There are three ways in which two component graphs can be combined:

join_graphs - The main concepts of both graphs must have a common semantic type (a common ancestor in the UMLS type hierarchy that is not too general). The type of the main concept of the new graph will be this common type, and the referent will be the union of referents of the main concepts of the two component graphs. The relations of the two graphs are then combined into a single list of relations.

relate_graphs - There must exist a UMLS semantic relation that can link the main concepts of the two component graphs. The main concept of the new graph is chosen to be the main concept of one of the two component graphs. The other graph is attached to this concept via the appropriate semantic relation.

append_graphs - The new graph is simply a concept with the two component graphs placed inside it. (This aggregate graph is called a 'context' in CG terminology).

The grammar shown in Figure 1 makes use of a restriction called **combine_graphs**, which is a generic version of the above three restrictions. **Combine_graphs** first attempts to find a way of orienting the two component graphs so that **join_graphs** applies. If this fails, the restriction tries to orient the graphs in such a way that **relate_graphs** applies. Failing these attempts, the graphs are combined using **append_graphs**.

RESULTS

AQUA is currently in a prototype stage, having been developed on a corpus of 158 queries (264 sentences) drawn from the NLM test collection. The purpose of the project in this phase of development is to evaluate the extent to which UMLS Knowledge Sources can be used to support natural language processing. The lexicon was built with the aid of the UMLS Metathesaurus. Of the 1609 lexical items in the corpus, 1209 (75%) were argument words. Only 761 (47%) of these could be assigned semantic types using automatic techniques to search the

Metathesaurus. The remaining lexical entries had to be constructed by hand.

AQUA required 481 distinct ways of linking semantic relations with pairs of semantic types to construct CG interpretations of the queries in the corpus. Of these, 215 links (45%) were found to already exist in the UMLS Semantic Net. Of the 134 semantic types present in the Semantic Net, 109 (81%) were actually used by AQUA. Of the 6227 instantiated triples possible in the Semantic Net, only 215 (3%) were required to interpret queries.

DISCUSSION

Although the number of semantic triples matching the Semantic Net may appear low, not all triples have equal weight. The most important and frequently occurring were present. Of those triples not found, a sizable portion was due transitive relationships not being represented directly. For example, there is no explicit triple in the net 'pharmacologic substance treats patient or disabled group'. However, the net does have 'disease or syndrome occurs in patient or disabled group' and 'pharmacologic substance treats disease or syndrome'. Such 'derived triples' can be generated from the Semantic Net automatically for use by the parser.

The corpus of user queries is helpful in establishing the breadth of topics in which users are interested, and to get some sense of the way in which information needs are expressed in natural language. The particular set of syntactic structures exhibited by the queries cannot be taken as definitive, since the corpus is made up of transcriptions of verbal requests. It is hypothesized that users will be much more terse when typing, and will adopt a simpler style when submitting requests to a computer than when addressing human mediators of their search. Moreover, users may adapt to the system over time, effectively learning the syntax of the implicit query language that the machine can interpret reliably.

CONCLUSION

AQUA has yet to undergo stringent evaluation. One interesting advantage of the current approach is that the system captures the user's stated information need, which can then be compared with the results of information retrieval. Even in the prototype stage it is clear that the declarative nature of DCG and the interactive Prolog environment facilitate rapid development of simple natural language interfaces. Operations on CG correspond naturally to the constraints required for query analysis, and were easily implemented in Prolog.

Evaluation of the UMLS clearly demonstrates that the Metathesaurus is not a substitute for a lexicon, particularly with regard to relation words such as verbs. If the UMLS is to support natural language processing, additional lexical information must be provided. Although not complete, the Semantic Net can easily be extended to provide the semantic links needed for query interpretation.

Acknowledgements

Support for this project was provided by a UMLS contract from the National Library of Medicine, and by the IBM Corporation.

References

[1] Obermeier KK. NLIs Lead the Way. DEC Professional 1988(May):70-8.

[2] Friedman C, Johnson SB. Medical Text Processing: Past Achievements, Future Directions. In: Ball MJ, Collen MF (eds). Aspects of the Computer-Based Record. New York: Springer-Verlag 1992.

[3] Pereira FC, Warren DH. Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks. Artificial Intelligence 1980(13):231-78.

[4] Sowa JF. Conceptual Graphs: information processing in mind and machine. Reading MA: Addison Wesley 1984.

[5] McCray AT, Hole WT: The Scope and Structure of the First Version of the UMLS Semantic Net. In: Miller RA, editor. Proc. of the 14th SCAMC. Washington: IEEE Computer Society Press, 1990:126-30.

[6] Allen J. Natural Language Understanding. Menlo Park CA: Benjamin/Cummings 1987:250-3.

[7] Sager N, Friedman C, Lyman MS. Medical Language Processing: Computer Management of Narrative Data. Reading MA: Addison Wesley 1987.

[8] Harris Z. A theory of language and Information. New York: Oxford University Press 1991.

[9] Johnson SB, Gottfried M. Sublanguage Analysis as a Basis for a Controlled Medical Vocabulary. In: Miller RA, editor. Proc. of the 13th SCAMC. Washington: IEEE Computer Society Press, 1989:519-23.